スマイル

SMILE

Smart Module with Instruction Language for Education

教育用プログラミングモジュール

SMILE1.00

取扱説明書

2008 年 6 月 29 日

東京都立日比谷高等学校　天良 和男（TENRA Kazuo）

目次

項目	項目名	ページ	項目	項目名	ページ
1	スマイル（SMILE）とは	4	6	プロパティとメソッド	16
2	スマイルの利用方法	5	6.1	プロパティ	16
2.1	インストールとアンインストール	5	6.1.1	「待ち時間」プロパティ	16
2.2	動作環境	5	6.1.2	「線の種類」プロパティ	17
2.3	セキュリティの設定	5	6.1.3	「線の太さ」プロパティ	18
2.3.1	Excel2003 の場合	5	6.1.4	「線の色」プロパティ	19
2.3.2	Excel2007 の場合	6	6.1.5	「楽器」プロパティ	19
2.4	スマイルの実行	6	6.1.6	「テンポ」プロパティ	20
2.4.1	Excel2003 の場合	6	6.1.7	「音量」プロパティ	21
2.4.2	Excel2007 の場合	6	6.1.8	「右」プロパティ	22
2.5	プログラムの実行	6	6.1.9	「左」プロパティ	23
2.5.1	Excel2003 の場合	6	6.1.10	「上」プロパティ	24
2.5.2	Excel2007 の場合	6	6.1.11	「下」プロパティ	25
2.6	プログラムの作成	7	6.1.12	「中心横」プロパティ	26
2.6.1	Excel2003 の場合	7	6.1.13	「中心縦」プロパティ	27
2.6.2	Excel2007 の場合	7	6.2	メソッド	29
3	タートルグラフィックス	7	6.2.1	「向きを変える」メソッド	29
3.1	タートルグラフィックスとは	7	6.2.2	「位置を移動する」メソッド	30
3.2	キャンバス	7	6.2.3	「位置を絶対移動する」メソッド	31
4	クラスとオブジェクト	8	6.2.4	「位置を相対移動する」メソッド	31
4.1	クラス	8	6.2.5	「色を変える」メソッド	32
4.2	オブジェクト	8	6.2.6	「大きさを変える」メソッド	33
5	オブジェクトの使用にあたって	8	6.2.7	「形を変える」メソッド	34
5.1	オブジェクト変数の宣言	8	6.2.8	「姿を消す」メソッド	34
5.2	オブジェクトの作成	9	6.2.9	「軌跡をグループ化する」メソッド	35
5.3	オブジェクトの使用	10	6.2.10	「軌跡を移動する」メソッド	36
5.3.1	コードの記述	11	6.2.11	「軌跡を塗りつぶす」メソッド	36
5.3.2	プロパティに値を設定	12	6.2.12	「キャラクタ名を変える」メソッド	37
5.3.3	変数にプロパティの値を取得	13	6.2.13	「三原色を変える」メソッド	38
5.3.4	メソッドの実行	14	6.2.14	「表情を変える」メソッド	38
5.4	オブジェクトの削除	15	6.2.15	「楽器を演奏する」メソッド	39

項目	項目名	ページ	項目	項目名	ページ
7	プロシージャ	43			
7.1	「キー入力判定」Function プロシージャ	43			
7.2	「待機」Sub プロシージャ	45			
7.3	「設定の窓表示」Sub プロシージャ	45			
7.4	「キャンバス」Sub プロシージャ	45			
7.5	「別ファイルに保存」Sub プロシージャ	46			
7.6	「別ファイルから読込」Sub プロシージャ	46			
7.7	「図形消去」Sub プロシージャ	47			
7.8	「透明化」Sub プロシージャ	47			
7.9	「音楽再生」Sub プロシージャ	47			
8	設定	48			
8.1	キャンパスの設定	48			
8.1.1	「横位置」スクロールバー	48			
8.1.2	「縦位置」スクロールバー	48			
8.1.3	「倍率」スクロールバー	49			
8.1.4	「位置 0　倍率 100」ボタン	49			
8.1.5	「キャンバスのセット」ボタン	49			
8.1.6	「キャンバスのリセット」ボタン	49			
8.1.7	「座標軸の目盛表示」ボタン	49			
8.1.8	「座標軸の目盛消去」ボタン	49			
8.2	別ファイルに保存	49			
8.2.1	エクスポート（コピー）	49			
8.2.2	エクスポート（移動）	50			
8.3	別ファイルから読み込み	50			
8.4	図形の削除	50			
8.5	ヘルプ	50			
8.6	サンプルプログラムについて	50			
8.7	スマイルについて	51			

スマイル

1　スマイル（SMILE）とは

・ SMILE の名称の語源

SMILE は，「Smart Module with Instruction Language for Education」の略称で，「教育用命令語を備えた知的モジュール」という意味をもちます。モジュールとは，取り外しや交換が簡単にできるプログラムや部品のことです。また, SMILE の基本機能の中に，タートルグラフィックスがあります。タートルグラフィックスとは，画面上のキャンバス上で，仮想の亀（タートル）に見立てたキャラクタに命令を与えて，移動させたり，向きを変えたりしながら，その軌跡を描くプログラムです。

SMILE のデフォルトのキャラクタは笑顔（SMILE）であり，それが本モジュールのもう一つの語源です。スマイルには笑顔以外にも，亀や様々な種類のポインタが用意されており，ユーザが独自に作成したキャラクタも登録できます。

・ SMILE は言語でなくモジュール

SMILE は EXCEL-VBA（Visual Basic for Applications）で開発されており，タートルグラフィックス機能や音楽機能をクラスモジュールとして開発し，それをアドイン化(xla)して，プログラムを記述するファイル(xls)から呼び出して利用します。アドインとは，ソフトウェアに追加される拡張機能のことです。SMILE は，プログラミング言語ではありません。SMILE に使われているプログラミング言語は Excel-VBA であり，その言語体系は，VisualBasic に準拠しており，利用度が高く初学者にわかりやすいといえます。アドイン化されたクラスモジュールだけでなく，通常の Excel－VBA がもっている機能をすべて利用できるため，教育用に特化しないで実用的なプログラミングが可能です。このようにスマイルは，VBA に教育用のプログラミング環境のための関数や命令群を追加した拡張クラスモジュールです。そのため Excel -VBA にスマイル拡張クラスモジュールを追加することで，従来の Excel-VBA によるプログラミングに加えて，タートルグラフィックスによる図形描画や MIDI（Musical Instrument Digital Interface）による音楽演奏などのプログラミングを容易に行うことができます。また，Excel のワークシートがタートルグラフィックスの座標系に対応しており，SMILE との組み合わせで，様々なプログラム作りが期待できます。

・ 視聴覚を刺激しながらプログラミング

SMILE の基本機能として，タートルグラフィックス機能と音楽演奏機能などがあります。タートルグラフィックス機能により，図形の描画を通して視覚を刺激しながらプログラミングができます。また，音楽演奏機能により，MIDI（Musical Instrument Digital Interface）音源の再生を通して聴覚を刺激しながらプログラミングができます。

・ 日本語表記

モジュール化されたクラスのプロパティやメソッドは，日本語表記されているため，プログラミングの初学者には分かりやすくなっています。

・ 描画した図形オブジェクトの利用

SMILE のタートルグラフィックス機能で描画した図形は，オートシェイプオブジェクトであるため，他の様々なソフトウェアにコピー&ペーストできます。

2 スマイルの利用方法

2.1 インストールとアンインストール

スマイルはレジストリーを使用していないので，インストールするには，スマイルの関連ファイルが入っている記録メディアから，フォルダごとハードディスクや USB フラッシュメモリ，MO ディスク等の読み書きできる記録メディアにコピーします。アンインストールは，フォルダごと削除することで実現できます。なお，フォルダ内に入っている「SmileProgram1.00.xls」，「SmileClass1.00.xla」，「oto1.wav」，「oto2.wav」，「追加プログラム 1.bas」，「Smile1.00.txt」は同一フォルダ内に入っている必要があります。

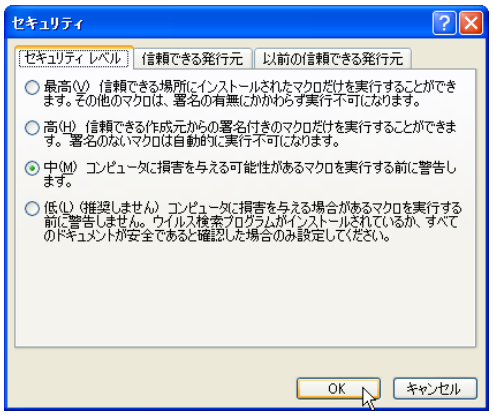
2.2 動作環境

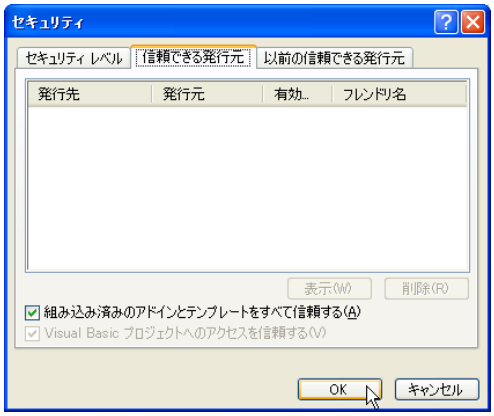
スマイルを動作させるには，OS として Windows XP または Windows Vista が，Excel として Excel 2003 または Excel 2007 がそれぞれインストールされている必要があります。

2.3 セキュリティの設定

2.3.1 Excel2003 の場合

スマイルを実行する場合，あらかじめセキュリティレベルを調整しておく必要があります。Excel を起動し，「ツール」→「マクロ」→「セキュリティ」を選択し，「セキュリティレベル」のタブを選択し「中」にします。次に「信頼できる発行元」のタブを選択し，「Visual Basic プロジェクトへのアクセスを信頼する」にチェックを付けます。「OK」をクリックし，いったん Excel を閉じます。スマイル実行中に，「プログラミングによる Visual-Basic プロジェクトへのアクセスは信頼性に欠けます。」というメッセージが出る場合は，「Visual Basic プロジェクトへのアクセスを信頼する」にチェックが付いていないことが原因ですので，必ずこの部分にチェックをつけてください。





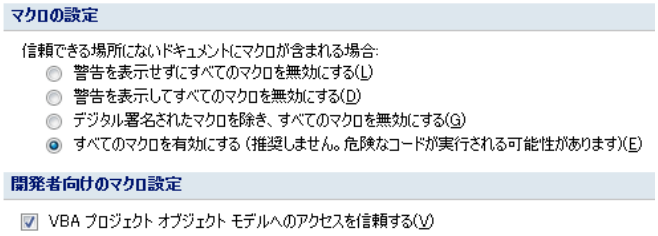
2.3.2　Excel2007，Excel2010 の場合

Excel2007 の場合は以下の①，③を，Excel2010 の場合は以下の②，③をそれぞれ操作してください。

①「Office ボタン」→「Excel のオプション」→「基本設定」カテゴリの「Excel の操作の基本オプション」で「[開発]タブをリボンに表示する」のチェックボックスにチェックを付ける→「OK」

②「ファイル」タブ→「オプション」→「リボンのユーザー設定」の「開発」のチェックボックスにチェックを付ける→「OK」

③「開発」タブの「コード」で「マクロセキュリティ」→「マクロの設定」カテゴリ→「すべてのマクロを有効にする（推奨しません。危険なコードが実行される可能性があります）→「VBA プロジェクトオブジェクトモデルへのアクセスを信頼する」のチェックボックスにチェックを付ける→OK→いったん Excel を閉じます。

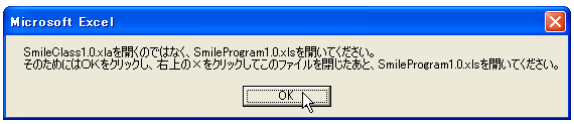


2.4　スマイルの実行

次に，「SmileProgram1.00.xls」をダブルクリックして開きます。

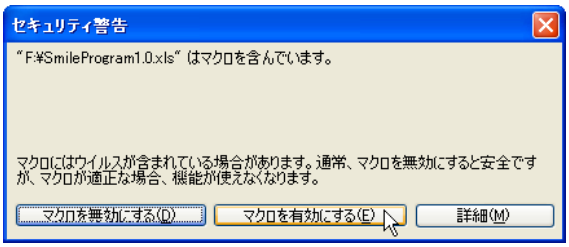
なお，「SmileClass1.00.xla」を開いてもスマイルを実行させることはできません。

間違って「SmileClass1.00.xla」を開いた場合は，いったん「SmileClass1.00.xla」を閉じてから，「SmileProgram1.00.xls」を開いてください。



2.4.1 Excel2003 の場合

「セキュリティ警告」の窓が開くので，「マクロを有効にする」のボタンをクリックします。



2.4.2 Excel2007，Excel2010 の場合

「セキュリティ警告」の窓は開きません。

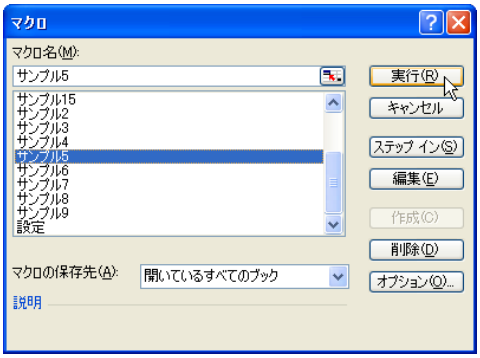
2.5　プログラムの実行

2.5.1 Excel2003 の場合

メニューバーからツール→マクロ→マクロ→マクロ名（プログラム名）を選択→実行

2.5.2 Excel2007，Excel2010 の場合

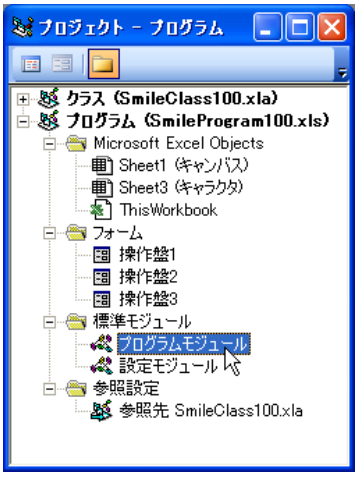
「開発」タブの「コード」→「マクロ」→マクロ名（プログラム名）を選択→実行



2.6 プログラムの作成

2.6.1 Excel2003 の場合

メニューバーから，「ツール」→「マクロ」→「Visual Basic Editor」を選択します。



2.6.2 Excel2007，Excel2010 の場合

「開発」タブの「コード」→「Visual Basic」を選択します。「Visual Basic Editor」が開くので，「プロジェクトエクスプローラ」の「標準モジュール」の中の「プログラムモジュール」をダブルクリックして，開いた窓の中にプログラムを記述します。「プロジェクトエクスプローラ」の窓が開いていない場合は，「表示」→「プロジェクトエクスプローラ」を選択すると表示します。

3 タートルグラフィックス

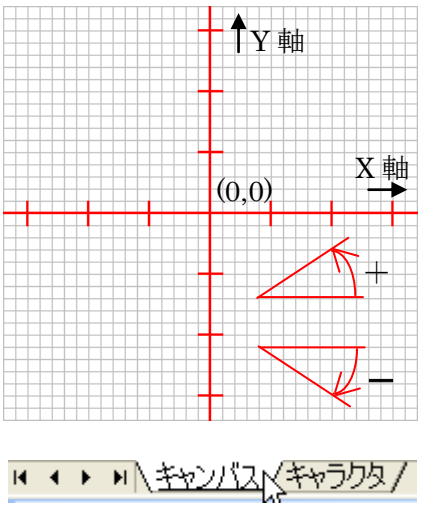
3.1 タートルグラフィックスとは

タートルグラフィックスとは，画面上のキャンバス上で，仮想の亀（タートル）に見立てたキャラクタに命令を与えて，移動させたり，向きを変えたりしながら，その軌跡を描くプログラムです。プログラミング言語「Logo」では描画用のキャラクタとして亀（タートル）が使われたところから，タートルグラフィックスと呼ばれています。

スマイル（SMILE）には，亀以外にも様々な種類のキャラクタがあり，笑顔（SMILE）も代表的なキャラクタの 1 つです。これがスマイルの名称の由来です。なお，ユーザが独自に作成したキャラクタも登録できます。

3.2 キャンバス

「キャンパス」シートは，描画用のキャラクタを用いて軌跡を描くことができる画面です。中心が原点で，横方向の X 軸と縦方向の Y 軸で座標を表しています。Excel の正方形のセル 1 個分が 1 目盛になっており，1 目盛の長さは 10 ピクセルに相当します。X と Y はともに-1280～+1280 の範囲内でキャラクタの位置を指定することができます。角度は X 軸の正方向が 0 度で，左回り（反時計回り）が正，右周り（時計回り）が負です。プログラムの実行は必ず「キャンパス」シートに切り替えて行ってください。



4　クラスとオブジェクト

4.1 クラス

何度でも使われる機能は，設計図としてひとつにまとめたプログラムにすれば，別のプログラムから再利用することができます。この設計図のプログラムをクラスと呼びます。

4.2 オブジェクト

ところがクラスだけでは，実際に動くプログラムにはなりません。設計図であるクラスを元にして作られた，実際に動くプログラムをオブジェクトといいます。オブジェクトにすれば，別のプログラムから再利用することができます。

オブジェクトにはプロパティとメソッドがあり，プロパティはオブジェクトの属性（性質や特徴，状態など）であり，メソッドはオブジェクトに動作を与える命令と考えるとよいでしょう。例えば，オブジェクトを「体」とすれば，プロパティは「体」の性質である「身長」や「体重」などを表しています。これに対してメソッドは「体」に「歩く」や「走る」などの動作を与える命令だと考えればわかりやすいでしょう。

5　オブジェクトの使用にあたって

オブジェクトを使用するときは，準備として，以下の説明における，5.1 のオブジェクト変数の宣言と，5.2 のオブジェクトの作成を行う必要があります。

次に，オブジェクトを使用するときは，以下の説明における 5.3 の 5.32～5.34 のコードを記述します。そのオブジェクトで公開されているプロパティとメソッドを使うことによってそのオブジェクトを操作することになります。

最後に，オブジェクトの使用が終了したら，5.4 のオブジェクトの削除を行います。

オブジェクトの削除を行わないと，コンピュータの動作中に，使用可能なメモリ容量がだんだん減っていく現象（メモリリーク）が起こります。

オブジェクトの準備（使用前）　5.1＋5.2
↓
オブジェクトの使用　5.3
↓
オブジェクトの削除（使用後）　5.4

5.1 オブジェクト変数の宣言

・ 書式

```
Dim オブジェクト変数名 As クラス名
```

・ 機能
オブジェクトを入れる変数を宣言します。
・ クラス名
スマイルには，図形用と音声用のクラスがあり，それぞれのクラス名は「図形クラス」，「音楽クラス」となっています。
・ オブジェクト変数名
オブジェクトを入れる変数に付ける名前で，Excel-VBA で使われている予約語やスマイルに使われている予約語と同じ名前以外ならば，どのような名前であってもかまいません。
・ 使用例 1

```
Dim 笑顔 As 図形クラス

「笑顔」という名前のオブジェクト変数を音楽クラスとして宣言します。
```

笑顔という名前のオブジェクト変数を図形クラスとして宣言するには，上記のように，「オブジェクト変数名」を「笑顔」とし，「クラス名」を「図形クラス」とします。

・ 使用例 2

```
Dim 奏者 As 音楽クラス

「奏者」という名前のオブジェクト変数を音楽クラスとして宣言します。
```

奏者という名前のオブジェクト変数を音楽クラスとして宣言するには，上記のように，「オブジェクト変数名」を「奏者」とし，「クラス名」を「音楽クラス」とします。

5.2 オブジェクトの作成

・ 書式

```
Set オブジェクト変数名 = オブジェクト作成用の関数名
```

・ 機能
オブジェクトを作成してオブジェクト変数に代入します。
・ オブジェクト作成用の関数名
オブジェクト作成用の関数には図形用と音楽用があり，それぞれの関数名は「図形オブ

ジェクト作成」,「音楽オブジェクト作成」となっています。

・ オブジェクト変数名
上記 5.1 で宣言したオブジェクト変数名と同じ変数名を使用します。

・ 使用例 1

```
Set 笑顔 = 図形オブジェクト作成
```

「笑顔」という名前の図形オブジェクトを作成します。

図形用のオブジェクトを作成して笑顔という名前のオブジェクト変数に代入するには,上記のように,「オブジェクト変数」を「笑顔」とし,「オブジェクト作成用の関数名」を「図形オブジェクト作成」とします。

・ 使用例 2

```
Set 奏者 = 音楽オブジェクト作成
```

「奏者」という名前の音楽オブジェクトを作成します。

音楽用のオブジェクトを作成して奏者という名前のオブジェクト変数に代入するには,上記のように,「オブジェクト変数」を「奏者」とし,「オブジェクト作成用の関数名」を「音楽オブジェクト作成」とします。

5.3 オブジェクトの使用

各オブジェクトにはプロパティとメソッドを使って操作を行います。
オブジェクトが持っているプロパティやメソッドはクラスの種類によって決まっています。これらのプロパティやメソッドの詳しい説明は 6 で行います。
プロパティ名には名詞が用いられ，メソッド名には動詞が用いられているので区別しやすいようになっています。図形クラスと音楽クラスのプロパティやメソッドにはそれぞれ次のようなものがあります。

・ 図形クラスのプロパティ
待ち時間，線の種類，線の太さ，線の色，右，左，上，下，中心横，中心縦

・ 音楽クラスのプロパティ
楽器　テンポ　音量

・ 図形クラスのメソッド
向きを変える，位置を移動する，位置を絶対移動する，位置を相対移動する

色を変える，大きさを変える，形を変える，姿を消す，軌跡をグループ化する
軌跡を移動する，軌跡を塗りつぶす，キャラクタ名を変える，三原色を変える
表情を変える

・ 音楽クラスのメソッド
楽器を演奏する

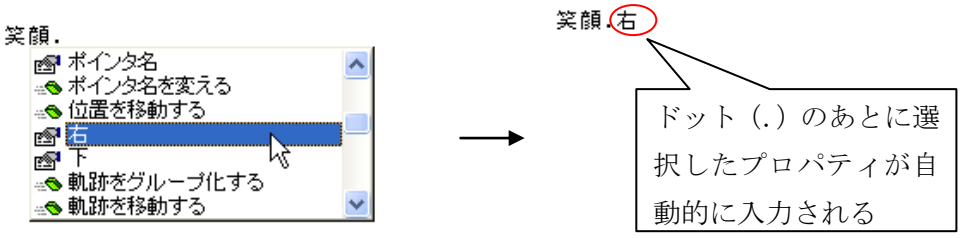
5.3.1 コードの記述

プロパティとメソッドのことをまとめて「メンバ」と呼び，コード記述時に「オブジェクト.」とすると，そのオブジェクトに関するプロパティやメソッドのメンバリストが表示されます。メンバリストに表示されるプロパティやメソッドのアイコンは以下の通りです。

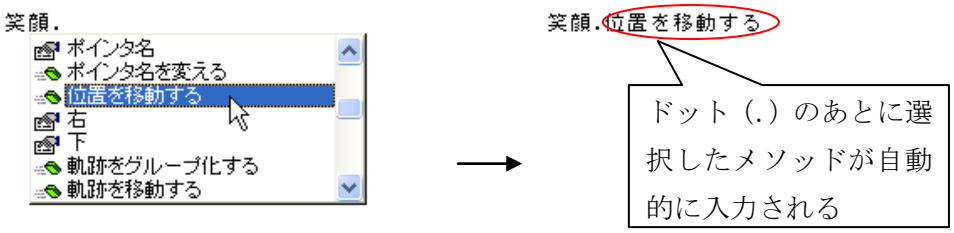
プロパティのアイコン

メソッドのアイコン

メンバリストに表示されるプロパティやメソッドをクリックするだけで，キーボードから入力しなくても自動的にこれらが記述されます。



図形クラスのプロパティを選択



図形クラスのメソッドを選択

プロパティを代入演算子「=」の左辺におくとプロパティに値を設定することができます。これに対して，プロパティを代入演算子「=」の右辺におくとプロパティに設定されている値を取得することができます。

5.3.2 プロパティに値を設定（プロパティを代入演算子「=」の左辺に置く）

・ 書式

オブジェクト変数名.プロパティ名 = 値

・ 機能

プロパティに値を設定します（右辺の値を左辺のプロパティに代入します)。

・ 値

プロパティに設定する値を代入演算子「＝」の右辺に記述します。

・ オブジェクト変数名

上記 5.1 で宣言したオブジェクト変数名と同じ変数名を使用します。

・ プロパティ名

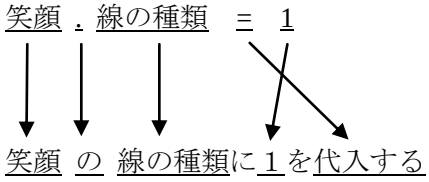
オブジェクト変数名に続けてドット（.）を記述し，そのあとにプロパティ名を記述して代入演算子「＝」の左辺に置きます。

・ 使用例 1

```
笑顔.線の種類 = 1
```

「笑顔」が描く軌跡を「矢印なしの直線」にします。

「笑顔」オブジェクトの「線の種類」プロパティに「1」を設定します。ここで「1」は「矢印なしの直線」をあらわしています。これにより，「笑顔」が描く軌跡は「矢印なしの直線」になります。なお，「笑顔」は図形クラスのオブジェクトである必要があります。詳しくは 6.12 をご覧ください。



ドット（.）は，日本語の「の」に置き換えると分かりやすいと思います。

・ 使用例 2

```
奏者.楽器 = 0
```

「奏者」が演奏する「楽器」を「グランドピアノ」にします。

「奏者」オブジェクトの「楽器」プロパティに「0」を設定します。ここで,「0」は「グランドピアノ」を表しています。これにより,「奏者」が演奏する「楽器」は「グランドピアノ」になります。なお,「奏者」は音楽クラスのオブジェクトである必要があります。詳しくは 6.15 をご覧ください。

奏者 . 楽器 = 0

奏者 の 楽器に 0 を代入する。

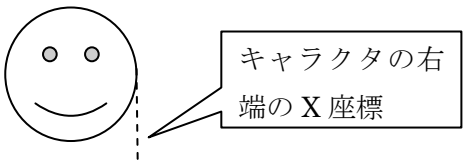
5.3.3 変数にプロパティの値を取得（プロパティを代入演算子「=」の右辺に置く）

・ 書式

変数名＝オブジェクト変数名.プロパティ名

・ 機能
変数にプロパティの値を取得します（右辺のプロパティを左辺の変数に代入します)。
・ オブジェクト変数名
上記 5.1 で宣言したオブジェクト変数名と同じ変数名を使用します。
・ プロパティ名
オブジェクト変数名に続けてドット（.）を記述し，そのあとにプロパティ名を記述して代入演算子「＝」の右辺に置きます。
・ 変数名
取得したプロパティの値を代入するための変数を代入演算子「＝」の左辺に置きます。変数にプロパティを代入せずに左辺の変数名と代入演算子「＝」を省略して「オブジェクト変数名.プロパティ名」としてプロパティを式の一部として取得することもできます。
・ 使用例

右座標 = 笑顔.右



「右座標」という名前の変数にキャラクタの右端の X 座標を代入します。

「笑顔」オブジェクトの「右」プロパティの値を,「右座標」という名前の変数に代入します。ここで,「右」プロパティの値はキャラクタの右端の X 座標を表しています。

これにより,「右座標」という名前の変数にキャラクタの右端の X 座標が代入されます。
なお,「笑顔」は図形クラスのオブジェクトである必要があります。
詳しくは 6.18 をご覧ください。

右座標　=　笑顔 . 右

右座標に笑顔 の 右を代入する

5.3.4 メソッドの実行

・ 書式

オブジェクト変数名.メソッド名 引数

または

Call　オブジェクト変数名.メソッド名 (引数)

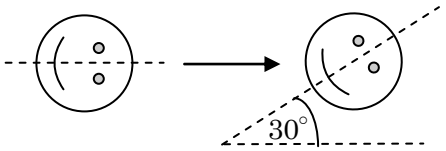
・ 機能
メソッドを実行します。引数がない場合は引数を省略します。
・ 注意事項
戻り値を持たないメソッドの呼び出しには，引数に括弧を付けないようにします。どうしても付けたいならば，上記のように Call を利用するようにします。

なお，以下のように，戻り値を持つメソッドの呼び出しには，引数に括弧を付け，その戻り値を受け取る変数を代入演算子「=」の左辺に置きますが，スマイルには戻り値を持つメソッドはありません。

変数名＝オブジェクト変数名.メソッド名 (引数)

・ オブジェクト変数名
上記 5.1 で宣言したオブジェクト変数名と同じ変数名を使用します。
・ メソッド名
オブジェクト変数名に続けてドット(.)を記述し,そのあとにメソッド名を記述します。
・ 引数
オブジェクトに引き渡す値です。引数がない場合は引数を省略します。
・ 使用例 1 （引数がある場合）
笑顔.向きを変える 30

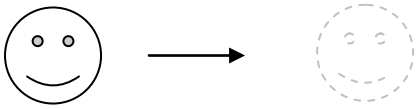
キャラクタの向きを左回りに 30 度変えます。



「笑顔」オブジェクトの「向きを変える」メソッドを実行します。ここで，引数「30」は左回りの角度を表しています。これにより，キャラクタの向きを左回りに 30 度変えます。なお,「笑顔」は図形クラスのオブジェクトでなければなりません。
また，この例は引数が 1 個の場合ですが，引数が複数個あるメソッドもあります。
詳しくは 6.2.1 をご覧ください。

・ 使用例２（引数がない場合）

笑顔.姿を消す



キャラクタを非表示にします。

「笑顔」オブジェクトの「姿を消す」メソッドを実行します。これにより，キャラクタは非表示になります。このメソッドを実行するとキャラクタは非表示になりますが，オブジェクトを削除したことにはなりません。なお,「笑顔」は図形クラスのオブジェクトである必要があります。詳しくは 6.2.8 をご覧ください。

5.4 オブジェクトの削除

・ 書式

```
Set オブジェクト変数名 = Nothing
```

・ 機能
オブジェクトを削除します。
・ オブジェクト変数名
上記 5.1 で宣言したオブジェクト変数名と同じ変数名を使用します。
・ 使用例

```
Set 笑顔 = Nothing
```

「笑顔」オブジェクトを削除します。

このメソッドを実行するとオブジェクトの削除とともにキャラクタも非表示になります。オブジェクトを削除すると，そのオブジェクトのプロパティやメソッドは使用できなくなります。なお,「笑顔」は図形クラスのオブジェクトである必要があります。

6　プロパティとメソッド

6.1 プロパティ

上記 5.3.2 および 5.3.3 で説明したように，一般的にプロパティには，値の設定だけできるもの，値の取得だけできるもの，値の設定と値の取得の両方ができるものがあります。ただし，スマイルには両方できるものはありません。

カッコ内は項目番号です。

- 値の設定だけができるプロパティ
  待ち時間(6.1.1)，線の種類(6.1.2)，線の太さ(6.1.3)，線の色(6.1.4)，楽器(6.1.5)，テンポ(6.1.6)，音量(6.1.7)
- 値の取得だけができるプロパティ
  右(6.1.8)，左(6.1.9)，上(6.1.10)，下(6.1.11)，中心横(6.1.12)，中心縦(6.1.13)

6.1.1 「待ち時間」プロパティ

- 書式

オブジェクト変数名.待ち時間 = 値

- 機能
  指定した時間だけ，プログラムの実行を中断します。値の設定のみ可能です。
  この機能を利用することで一定時間間隔でプログラムを実行できます。キャラクタの動きの速度が速すぎると描画の順序が分かりにくくなります。そこで，このプロパティで待ち時間を与えることによって描画速度を遅くします。
  例えば，キャラクタの向きを右回りに 30 度変える動作を「動作１」とし，キャラクタの中心を 50 移動させる動作を「動作２」とすると，「動作１」と「動作２」の間に「待ち時間」プロパティで設定した時間だけプログラムの実行を中断します。
  これは「動作１」と「動作２」の間だけでなく，次のようにプログラムの終了まで各動作の間に「待ち時間」で設定した時間が挿入されます。

  プログラムの開始 動作1, 待ち時間 動作2, 待ち時間 動作3 -――― 待ち時間 動作n－1, 待ち時間 動作n, プログラムの終了

- 値 「長整数型 (Long)」
  待ち時間をミリ秒（ms）単位で与えます。代入演算子「=」の右辺に置きます。
  このプロパティを使用しない場合は，待ち時間が自動的に 0 に設定されます。
- オブジェクト変数名
  オブジェクト変数は図形クラスのオブジェクトである必要があります。
- 注意事項
  音楽クラスのオブジェクトはこのプロパティを持っておりません。
  複数のオブジェクトに別々の待ち時間を与えることはできません。あるオブジェクト

に設定した待ち時間は，他のオブジェクトにも同じ待ち時間が反映されます。
「待機」Sub プロシージャも指定した時間だけプログラムの実行を中断します。
「待ち時間」プロパティはプログラムの終了まで各動作の間に「待ち時間」で設定した時間が挿入されますが，「待機」Sub プロシージャは，プロシージャを実行した時点だけプログラムの実行を中断します。詳しくは 7.2 をご覧ください。

・ 使用例

```
笑顔.待ち時間 = 500
```

「笑顔」が描く軌跡の線を「500 ミリ秒」間隔で描画します。

「笑顔」オブジェクトの「待ち時間」プロパティに「500」を設定します。ここで「500」は「500 ミリ秒（＝0.5 秒）」をあらわしています。これにより，「笑顔」の動きが「500 ミリ秒」間隔で反映されます。なお，「笑顔」は図形クラスのオブジェクトである必要があります。
なお，複数のオブジェクトにそれぞれ異なる待ち時間を設定することができません。例えば，「笑顔 1」と「笑顔 2」の２つの図形クラスのオブジェクトがあった場合に，「笑顔 1.待ち時間 = 500」のあとに，「笑顔 2.待ち時間 = 1000」を設定すると，その待ち時間は次のようになります。
設定した設定値はすべての図形オブジェクトに反映されるため，「笑顔 1.待ち時間 = 500」の設定直後は 500 ミリ秒が「笑顔１」と「笑顔２」の両方のオブジェクトに反映され，「笑顔 2.待ち時間 = 1000」の設定直後は「1000 ミリ秒（＝１秒）」の待ち時間が，両方のオブジェクトに反映されます。

6.1.2 「線の種類」プロパティ

・ 書式

```
オブジェクト変数名.線の種類 = 値
```

・ 機能
キャラクタが描く線の種類を設定します。値の設定のみ可能です。

・ 値「整数型 (Integer)」
以下のような線の種類を表す 0～255 の整数を設定します。
代入演算子「=」の右辺に置きます。
0：線なし，1：矢印なしの直線，2：矢印付きの直線，3～255 の整数：フリーフォーム
フリーフォームはクリックするたびに頂点が生まれますが，ドラッグすると手書き風

の線を描くことができます。
なお，このプロパティを使用しない場合は，自動的に 2 が設定されます。

・ オブジェクト変数名
オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ 使用例

```
笑顔.線の種類 = 1
```

「笑顔」が描く軌跡の線の種類を「矢印なしの直線」にします。

「笑顔」オブジェクトの「線の種類」プロパティに「1」を設定します。ここで「1」は「矢印なしの直線」をあらわしています。これにより，「笑顔」が描く軌跡の線の種類は「矢印なしの直線」になります。なお，「笑顔」は図形クラスのオブジェクトである必要があります。

6.1.3 「線の太さ」プロパティ

・ 書式

```
オブジェクト変数名.線の太さ = 値
```

・ 機能
オブジェクトが描く線の太さを設定します。
値の設定のみ可能です。

・ 値「単精度浮動小数点数型 (Single)」
線の太さを表す値を単精度浮動小数点型のポイント(Point)単位で設定します。
代入演算子「=」の右辺に置きます。
1 ポイントは 1/72 インチ，約 0.35 mm。
なお，このプロパティを使用しない場合は，2 が設定されます。

・ オブジェクト変数名
オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ 使用例

```
笑顔.線の太さ = 3
```

「笑顔」が描く軌跡の線の太さを「3 ポイント」にします。

「笑顔」オブジェクトの「線の太さ」プロパティに「3」を設定します。ここで「3」は「3 ポイント」をあらわしています。これにより,「笑顔」が描く軌跡の線の太さは「3 ポイント」になります。なお,「笑顔」は図形クラスのオブジェクトである必要があります。

6.1.4 「線の色」プロパティ

・ 書式

```
オブジェクト変数名.線の色 = 値
```

・ 機能

オブジェクトが描く線の色を設定します。

値の設定のみ可能です。

・ 値「整数型 (Integer)」

線の色を表す値を設定します。代入演算子「=」の右辺に置きます。

例

8：黒，9:白，10：赤，11：緑，12:青，13:黄，14:マゼンタ，15:シアン

なお，このプロパティを使用しない場合は，64（自動，黒）が設定されます。

・ オブジェクト変数名

オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ 使用例

```
笑顔.線の色 = 10
```

「笑顔」が描く軌跡の線の色を「赤色」にします。

「笑顔」オブジェクトの「線の色」プロパティに「10」を設定します。ここで「10」は「赤色」をあらわしています。これにより,「笑顔」が描く軌跡の線の色は「赤色」になります。なお,「笑顔」は図形クラスのオブジェクトである必要があります。

6.1.5 「楽器」プロパティ

・ 書式

```
オブジェクト変数名.楽器 = 値
```

・ 機能

オブジェクトが演奏する楽器の種類を設定します。
値の設定のみ可能です。音楽用メソッドについては 6.2.13 をご覧ください。

・ 値「長整数型 (Long)」
楽器の種類を表す値（0～127 の整数）を設定します。
代入演算子「=」の右辺に置きます。
値の例（その他の楽器は 6.2.13 の MIDI 楽器の種類と番号の表を参照してください。）
0：グランドピアノ，10:オルゴール，22：ハーモニカ，40:バイオリン，
56:トランペット，71：クラリネット，73:フルート，107:箏，116:和太鼓
なお，このプロパティを使用しない場合は，0（グランドピアノ）が設定されます。

・ オブジェクト変数名
オブジェクト変数は音楽クラスのオブジェクトである必要があります。

・ 使用例

```
奏者.楽器 = 40
```

「奏者」が演奏する楽器の種類を「バイオリン」にします。

「奏者」オブジェクトの「楽器」プロパティに「40」を設定します。ここで「40」は「バイオリン」をあらわしています。これにより,「奏者」が演奏する楽器の種類は「バイオリン」になります。なお,「奏者」は音楽クラスのオブジェクトである必要があります。

6.1.6 「テンポ」プロパティ

・ 書式

```
オブジェクト変数名.テンポ = 値
```

・ 機能
オブジェクトが演奏する楽器のテンポを設定します。
値の設定のみ可能です。音楽用メソッドについては 6.2.13 をご覧ください。

・ 値「長整数型 (Long)」
テンポを表す値（整数）を設定します。
代入演算子「=」の右辺に置きます。
なお，このプロパティを使用しない場合は，120 が設定されます。
TEMPO とは，1 分間に何拍四分音符があるかをあらわします。たとえば，TEMPO=120 は 1 分間に 120 拍四分音符があることになり，四分音符の時間の長さは 60 秒／120＝

0.5 秒＝500 ミリ秒になります。八分音符ではこの値に 1/2 を掛ければよいので，TEMPO120 の場合の八分音符の時間の長さは 250 ミリ秒になります。

・ オブジェクト変数名
オブジェクト変数は音楽クラスのオブジェクトである必要があります。

使用例

```
奏者.テンポ = 60
```

「奏者」が演奏する楽器のテンポを「60」にします。

「奏者」オブジェクトの「テンポ」プロパティに「60」を設定します。ここで「60」は 1 分間に 60 拍の四分音符があることをあらわしています。これにより，「奏者」が演奏する楽器のテンポは「60」になります。なお，「奏者」は音楽クラスのオブジェクトである必要があります。

6.1.7 「音量」プロパティ

・ 書式

オブジェクト変数名.音量 = 値

・ 機能
オブジェクトが演奏する楽器の音量を設定します。
値の設定のみ可能です。音楽用メソッドについては 6.2.13 をご覧ください。

・ 値「長整数型 (Long)」
楽器の音量を表す値（0～127 の整数）を設定します。
代入演算子「=」の右辺に置きます。
なお，このプロパティを使用しない場合は，127 が設定されます。

・ オブジェクト変数名
オブジェクト変数は音楽クラスのオブジェクトである必要があります。

・ 使用例

```
奏者.音量 = 60
```

「奏者」が演奏する楽器の音量を「60」にします。

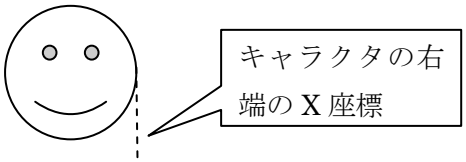
「奏者」オブジェクトの「音量」プロパティに「60」を設定します。なお，「奏者」は

音楽クラスのオブジェクトである必要があります。

6.1.8 「右」プロパティ

・ 書式

変数名 = オブジェクト変数名.右



・ 機能

オブジェクトのキャラクタの右端の X 座標を取得します。

値の取得のみ可能です。

・ オブジェクト変数名

オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ プロパティの値の型，変数名

プロパティの値の型は単精度浮動小数点数型 (Single)であるため，この変数も同じ型を使用し，代入演算子「=」の左辺に置きます。

なお，変数にプロパティを代入せずにプロパティの値を式の一部として取得することもできます。

・ 使用例 1

```
Dim 右座標 As Single
右座標 = 笑顔.右
```

「右座標」という名前の変数にキャラクタの右端の X 座標を代入します。

「笑顔」オブジェクトの「右」プロパティの値を，「右座標」という名前の変数に代入します。「=」は代入演算子です。ここで，「右」プロパティの値はキャラクタの右端のX座標を表しています。これにより，「右座標」という名前の変数にキャラクタの右端の X 座標が代入されます。なお，「笑顔」は図形クラスのオブジェクトである必要があります。

・ 使用例 2

```
If 笑顔.右 = 100 Then
        MsgBox "あたり"
End If
```

キャラクタの右端のX座標が 100 と等しいならば，「あたり」を表示します。

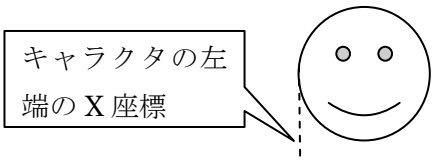
この例は，変数にプロパティを代入せずに，プロパティの値を条件式「笑顔.右 = 100」の一部として取得しています。

If Then ステートメントの条件式に使われている「=」は代入演算子ではなく，比較演算子です。よって「笑顔.右 = 100」は，100 を「笑顔」オブジェクトの「右」プロパティに代入するのではなく，「笑顔」オブジェクトの「右」プロパティの値が 100 に等しいという意味になります。

6.1.9 「左」プロパティ

- 書式

変数名 = オブジェクト変数名.左



- 機能

  オブジェクトのキャラクタの左端の X 座標を取得します。

  値の取得のみ可能です。
- オブジェクト変数名

  オブジェクト変数は図形クラスのオブジェクトである必要があります。
- プロパティの値の型，変数名

  プロパティの値の型は単精度浮動小数点数型 (Single)であるため，この変数も同じ型を使用し，代入演算子「=」の左辺に置きます。

  なお，変数にプロパティを代入せずにプロパティの値を式の一部として取得することもできます。
- 使用例 1

```
Dim 左座標 As Single
左座標 = 笑顔.左
```

「左座標」という名前の変数にキャラクタの左端の X 座標を代入します。

「笑顔」オブジェクトの「左」プロパティの値を，「左座標」という名前の変数に代入します。「=」は代入演算子です。ここで，「左」プロパティの値はキャラクタの左端の X 座標を表しています。これにより，「左座標」という名前の変数にキャラクタの左端の X 座標が代入されます。なお，「笑顔」は図形クラスのオブジェクトである必要があります。

- 使用例 2

```
If 笑顔.左 = 100 Then
        MsgBox "あたり"
End If
```

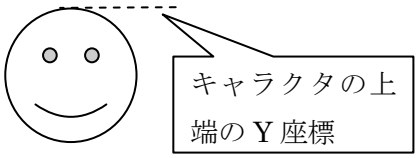
キャラクタの左端のX座標が 100 と等しいならば,「あたり」を表示します。

この例は，変数にプロパティを代入せずに，プロパティの値を条件式「笑顔.左 = 100」の一部として取得しています。
If Then ステートメントの条件式に使われている「=」は代入演算子ではなく，比較演算子です。よって「笑顔.左 = 100」は，100 を「笑顔」オブジェクトの「左」プロパティに代入するのではなく,「笑顔」オブジェクトの「左」プロパティの値が 100 に等しいという意味になります。

6.1.10 「上」プロパティ

- 書式

```
変数名 = オブジェクト変数名.上
```



- 機能
  オブジェクトのキャラクタの上端の Y 座標を取得します。
  値の取得のみ可能です。
- オブジェクト変数名
  オブジェクト変数は図形クラスのオブジェクトである必要があります。
- プロパティの値の型，変数名
  プロパティの値の型は単精度浮動小数点数型 (Single)であるため，この変数も同じ型を使用し，代入演算子「=」の左辺に置きます。
  なお，変数にプロパティを代入せずにプロパティの値を式の一部として取得することもできます。
- 使用例 1

```
Dim 上座標 As Single
上座標 = 笑顔.上
```

「上座標」という名前の変数にキャラクタの上端の Y 座標を代入します。

「笑顔」オブジェクトの「上」プロパティの値を,「上座標」という名前の変数に代入

します。「=」は代入演算子です。ここで,「上」プロパティの値はキャラクタの上端の Y 座標を表しています。これにより,「上座標」という名前の変数にキャラクタの上端の Y 座標が代入されます。なお,「笑顔」は図形クラスのオブジェクトである必要があります。

・ 使用例 2

```
If 笑顔.上 = 100 Then
        MsgBox "あたり"
End If
```

キャラクタの上端の Y 座標が 100 と等しいならば,「あたり」を表示します。

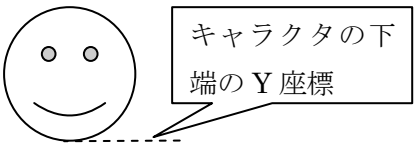
この例は，変数にプロパティを代入せずに，プロパティの値を条件式「笑顔.上 = 100」の一部として取得しています。

If Then ステートメントの条件式に使われている「=」は代入演算子ではなく，比較演算子です。よって「笑顔.上 = 100」は，100 を「笑顔」オブジェクトの「上」プロパティに代入するのではなく,「笑顔」オブジェクトの「上」プロパティの値が 100 に等しいという意味になります。

6.1.11 「下」プロパティ

・ 書式

```
変数名 = オブジェクト変数名.下
```



・ 機能

オブジェクトのキャラクタの下端の Y 座標を取得します。

値の取得のみ可能です。

・ オブジェクト変数名

オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ プロパティの値の型，変数名

プロパティの値の型は単精度浮動小数点数型 (Single)であるため，この変数も同じ型を使用し，代入演算子「=」の左辺に置きます。

なお，変数にプロパティを代入せずにプロパティの値を式の一部として取得することもできます。

・ 使用例 1

```
Dim 下座標 As Single
下座標 = 笑顔.下
```

「下座標」という名前の変数にキャラクタの下端の Y 座標を代入します。

「笑顔」オブジェクトの「下」プロパティの値を,「下座標」という名前の変数に代入します。「=」は代入演算子です。ここで,「下」プロパティの値はキャラクタの下端の Y 座標を表しています。これにより,「下座標」という名前の変数にキャラクタの下端の Y 座標が代入されます。なお,「笑顔」は図形クラスのオブジェクトである必要があります。

・ 使用例 2

```
If 笑顔.下 = 100 Then
        MsgBox "あたり"
End If
```

キャラクタの下端の Y 座標が 100 と等しいならば,「あたり」を表示します。

この例は，変数にプロパティを代入せずに，プロパティの値を条件式「笑顔.下 = 100」の一部として取得しています。

If Then ステートメントの条件式に使われている「=」は代入演算子ではなく，比較演算子です。よって「笑顔.下 = 100」は，100 を「笑顔」オブジェクトの「下」プロパティに代入するのではなく,「笑顔」オブジェクトの「下」プロパティの値が 100 に等しいという意味になります。

6.1.12 「中心横」プロパティ

・ 書式

```
変数名 = オブジェクト変数名.中心横
```

・ 機能

オブジェクトのキャラクタの中心の X 座標を取得します。

値の取得のみ可能です。

・ オブジェクト変数名

オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ プロパティの値の型，変数名

プロパティの値の型は単精度浮動小数点数型 (Single)であるため，この変数も同じ型を使用し，代入演算子「=」の左辺に置きます。
なお，変数にプロパティを代入せずにプロパティの値を式の一部として取得することもできます。

・ 使用例 1

```
Dim 中心横座標 As Single
中心横座標 = 笑顔.中心横
```

「中心横座標」という名前の変数にキャラクタの中心の X 座標を代入します。

「笑顔」オブジェクトの「中心横」プロパティの値を，「中心横座標」という名前の変数に代入します。「=」は代入演算子です。ここで，「中心横座標」プロパティの値はキャラクタの中心のX座標を表しています。これにより，「中心横座標」という名前の変数にキャラクタの中心の X 座標が代入されます。なお，「笑顔」は図形クラスのオブジェクトである必要があります。

・ 使用例 2

```
If 笑顔.中心横 = 100 Then
        MsgBox "あたり"
End If
```

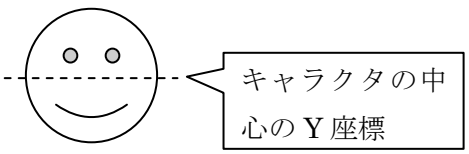
キャラクタの中心のX座標が 100 と等しいならば，「あたり」を表示します。

この例は，変数にプロパティを代入せずに，プロパティの値を条件式「笑顔.中心横 = 100」の一部として取得しています。
If Then ステートメントの条件式に使われている「=」は代入演算子ではなく，比較演算子です。よって「笑顔. 中心横= 100」は，100 を「笑顔」オブジェクトの「中心横」プロパティに代入するのではなく，「笑顔」オブジェクトの「中心横」プロパティの値が 100 に等しいという意味になります。

6.1.13 「中心縦」プロパティ

・ 書式

```
変数名 = オブジェクト変数名.中心縦
```



・ 機能

オブジェクトのキャラクタの中心の Y 座標を取得します。
値の取得のみ可能です。

- オブジェクト変数名
  オブジェクト変数は図形クラスのオブジェクトである必要があります。
- プロパティの値の型，変数名
  プロパティの値の型は単精度浮動小数点数型 (Single)であるため，この変数も同じ型を使用し，代入演算子「=」の左辺に置きます。なお，変数にプロパティを代入せずにプロパティの値を式の一部として取得することもできます。
- 使用例 1

```
Dim 中心縦座標 As Single
中心縦座標 = 笑顔.中心縦
```

「中心縦座標」という名前の変数にキャラクタの中心の Y 座標を代入します。

「笑顔」オブジェクトの「中心縦」プロパティの値を，「中心縦座標」という名前の変数に代入します。「=」は代入演算子です。ここで，「中心縦座標」プロパティの値はキャラクタの中心の Y 座標を表しています。これにより，「中心縦座標」という名前の変数にキャラクタの中心の Y 座標が代入されます。なお，「笑顔」は図形クラスのオブジェクトである必要があります。

- 使用例 2

```
If 笑顔.中心縦 = 100 Then
        MsgBox "あたり"
End If
```

キャラクタの中心の Y 座標が 100 と等しいならば，「あたり」を表示します。

この例は，変数にプロパティを代入せずに，プロパティの値を条件式「笑顔.中心縦 = 100」の一部として取得しています。If Then ステートメントの条件式に使われている「=」は代入演算子ではなく，比較演算子です。よって「笑顔. 中心縦= 100」は，100 を「笑顔」オブジェクトの「中心縦」プロパティに代入するのではなく，「笑顔」オブジェクトの「中心縦」プロパティの値が 100 に等しいという意味になります

6.2 メソッド

メソッドはオブジェクトに対して行う様々な操作をいいます。上記 5.3 で説明したように，図形クラスと音楽クラスにはそれぞれ以下のメソッドがあります。カッコ内は項目番号です。

- 図形クラスのメソッド

  向きを変える(6.2.1)，位置を移動する(6.2.2)，座標を絶対移動する(6.2.3)，位置を相対移動する(6.2.4)，色を変える(6.2.5)，大きさを変える(6.2.6)，形を変える(6.2.7)，姿を消す(6.2.8), 軌跡をグループ化する(6.2.9), 軌跡を移動する(6.2.10), 軌跡を塗りつぶす(6.2.11), キャラクタ名を変える(6.2.12) ，三原色を変える(6.2.13)，表情を変える(6.2.14)

- 音楽クラスのメソッド

  楽器を演奏する(6.2.15)

6.2.1 「向きを変える」メソッド

- 書式

  オブジェクト変数名.向きを変える 角度

- 機能

  キャラクタの向きを指定した角度だけ変えます。

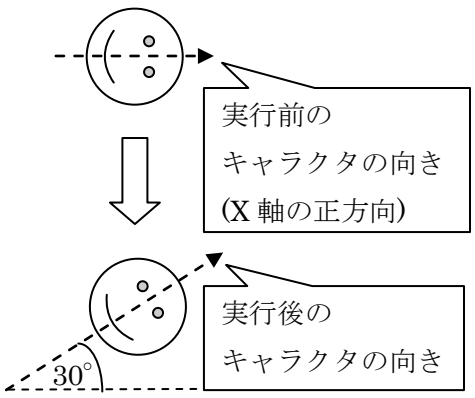
  「向きを変える」メソッドを一度も実行しない場合の向きはX軸の正方向です。

- オブジェクト変数名

  オブジェクト変数は図形クラスのオブジェクトである必要があります。

- 角度「単精度浮動小数点数型 (Single)」

  角度は向きを変える角度の値をあらわします。キャラクタの回転角度を現在の角度からどのくらい変更するかを指定します。角度は X 軸の正方向が 0 度で，左回り（反時計回り）が正，右周り（時計回り）が負です（3.2 参照)。



- 使用例 1

  笑顔.向きを変える 30

  キャラクタの向きを左回りに 30 度変えます。

  「笑顔」オブジェクトの「向きを変える」メソッドを実行します。ここで，引数「30」は左回りの角度を表しています。これにより，キャラクタの向きを左回りに 30 度変えます。なお,「笑顔」は図形クラスのオブジェクトでなければなりません。

6.2.2 「位置を移動する」メソッド

・ 書式

オブジェクト変数名.位置を移動する 移動量

・ 機能

キャラクタの中心を指定した移動量だけ移動します。

キャラクタの向きは,「向きを変える」メソッドで指定した向きです。「向きを変える」メソッドを一度も実行しない場合の移動する向きはX軸の正方向です。

・ オブジェクト変数名

オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ 移動量「単精度浮動小数点数型 (Single)」

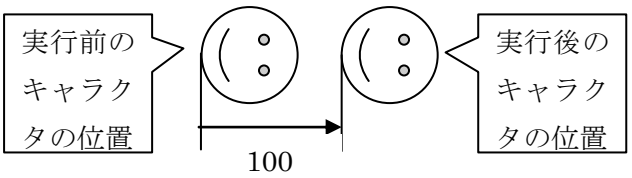
移動量は移動する長さの数値をあらわします。

この値を正にするとキャラクタの向きに前進し，負にすると後退（キャラクタの向きとは逆向きに進む）します。

・ 使用例 1

```
笑顔.位置を移動する 100
```

キャラクタの中心を100移動します。



「向きを変える」メソッドを一度も実行しないで,
「笑顔.位置を移動する 100」を実行した場合

「笑顔」オブジェクトの「位置を移動する」メソッドを実行します。ここで, 引数「100」は移動する量を表しています。これにより，キャラクタの位置を 100 移動します。

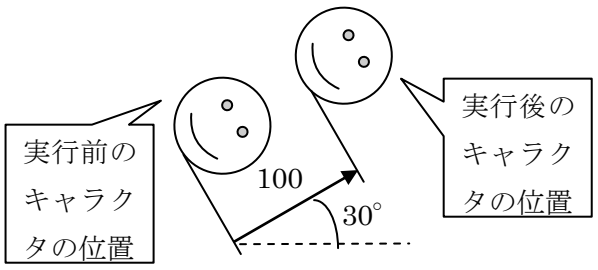
なお,「笑顔」は図形クラスのオブジェクトでなければなりません。

・ 使用例 2

```
笑顔.向きを変える 30
笑顔.位置を移動する 100
```

キャラクタの向きを左回りに 30 度変えます。

キャラクタの中心を 100 移動します。



「笑顔.向きを変える 30」を実行したあと,
「笑顔.位置を移動する 100」を実行した場合

6.2.3 「位置を絶対移動する」メソッド

・ 書式

オブジェクト変数名.位置を絶対移動する X 座標,Y 座標

・ 機能

キャラクタの中心の縦軸座標を X 座標の位置に，横軸座標を Y 座標の位置にそれぞれ移動します。

・ オブジェクト変数名

オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ X 座標「単精度浮動小数点数型 (Single)」，Y 座標「単精度浮動小数点数型 (Single)」

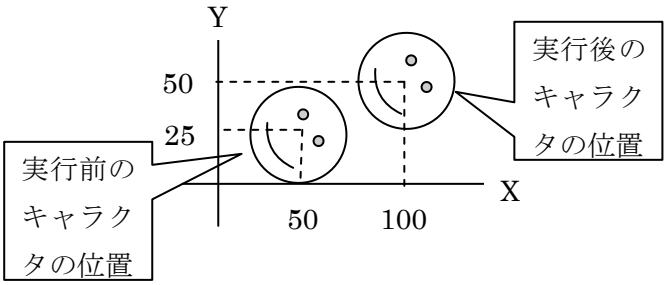
X 座標は移動先の横軸座標，Y 座標は移動先の縦軸座標をあらわします。座標は項目 3.2 を参照してください。

・ 使用例

笑顔.位置を絶対移動する 100,50

キャラクタの中心を X 座標=100 の位置に,Y 座標=50 の位置にそれぞれ移動します。

「笑顔」オブジェクトの「位置を絶対移動する」メソッドを実行します。ここで，１つ目の引数「100」は X 座標を，２つ目の引数「50」は Y 座標を表しています。これにより，キャラクタを X 座標が 100 の位置に,Y 座標が 50 の位置にそれぞれ移動します。図は，このメソッドを実行する前のキャラクタの座標が (50, 25) の位置にあった場合を表しています。なお,「笑顔」は図形クラスのオブジェクトでなければなりません。



6.2.4 「位置を相対移動する」メソッド

・ 書式

オブジェクト変数名.位置を相対移動する X 方向の移動量,Y 方向の移動量

・ 機能

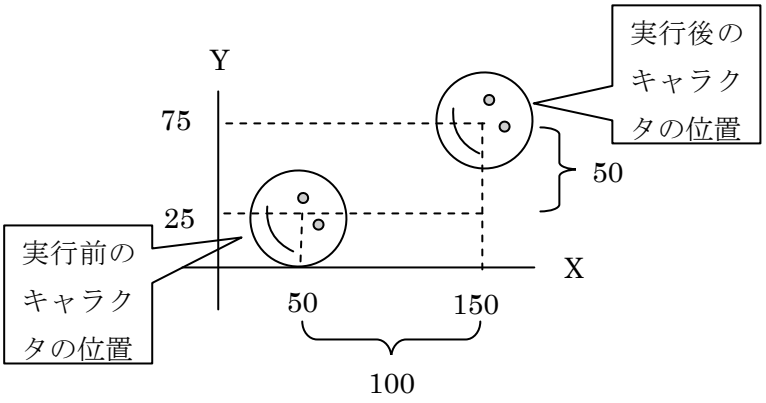
キャラクタの中心を横軸方向に X 方向の移動量,縦軸方向に Y 方向の移動量だけ，それぞれ移動します。

・ オブジェクト変数名
オブジェクト変数は図形クラスのオブジェクトである必要があります。
・ X 方向の移動量「単精度浮動小数点数型 (Single)」
Y 方向の移動量「単精度浮動小数点数型 (Single)」
X 方向の移動量は横軸方向に移動する量を，Y 方向の移動量は縦軸方向に移動する量をそれぞれあらわしています。
X 方向の移動量が負の場合は, X 軸の負の方向に移動し, Y 方向の移動量が負の場合は, Y 軸の負の方向に移動します。座標は項目 3.2 を参照してください。
・ 使用例

笑顔.位置を相対移動する 100,50

キャラクタを X 方向に 100，Y 方向に 50 それぞれ移動します。

「笑顔」オブジェクトの「位置を相対移動する」メソッドを実行します。ここで，１つ目の引数「100」は X 方向に移動する量を，２つ目の引数「50」は Y 方向に移動する量を表しています。図は，このメソッドを実行する前のキャラクタの位置座標が（50，25）にあった場合を表しています。



なお,「笑顔」は図形クラスのオブジェクトでなければなりません。

6.2.5 「色を変える」メソッド
・ 書式

オブジェクト変数名.色を変える 色

・ 機能
キャラクタの色を指定した色に変えます。
・ オブジェクト変数名
オブジェクト変数は図形クラスのオブジェクトである必要があります。
・ 色「整数型 (Integer)」

色はキャラクタの色をあらわします。

例

8：黒，9:白，10：赤，11：緑，12:青，13:黄，14:マゼンタ，15:シアン

なお，このプロパティを使用しない場合は，65（自動，白）が設定されます。

・ 使用例

笑顔.色を変える 10

キャラクタの色を10（赤）に変えます。

「笑顔」オブジェクトの「色を変える」メソッドを実行します。ここで，引数「10」は赤色を表しています。

なお,「笑顔」は図形クラスのオブジェクトでなければなりません。

6.2.6 「大きさを変える」メソッド

・ 書式

オブジェクト変数名.大きさを変える 倍率

・ 機能

キャラクタの大きさを変えます。

・ オブジェクト変数名

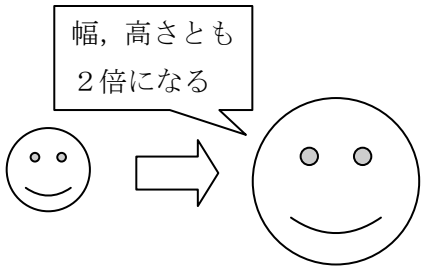
オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ 倍率「単精度浮動小数点数型 (Single)」

倍率は元の大きさに対する変化後の大きさの比をあらわします。

縦横の比率を変えないで，変更後のキャラクタの幅（高さ）と現在のキャラクタの幅（高さ）の比率を指定します。

・ 使用例



笑顔.大きさ変える 2

キャラクタを2倍の大きさに変えます。

「笑顔」オブジェクトの「大きさを変える」メソッドを実行します。ここで，引数「2」は倍率を表しています。

なお,「笑顔」は図形クラスのオブジェクトでなければなりません。

6.2.7 「形を変える」メソッド

・ 書式

オブジェクト変数名.形を変える キャラクタ名

・ 機能

キャラクタを指定したキャラクタ名のキャラクタに変えます。

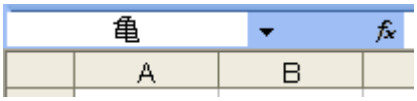
・ オブジェクト変数名

オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ キャラクタ名「文字列型 (String)」

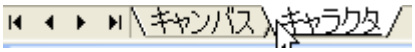
キャラクタ名は変更するキャラクタの名前をあらわします。

「キャラクタ」シートにある図形の名前を設定します。



「キャラクタ」シート上の図形をクリックすると，名前ボックスに図形の名前が表示されます。この名前をキャラクタ名に指定します。

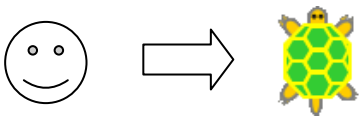
プログラムの実行は必ず「キャラクタ」シートから「キャンパス」シートに切り替えて行ってください。



・ 使用例

笑顔.形を変える "亀"

キャラクタの形を「亀」に変えます。



・ 「笑顔」オブジェクトの「形を変える」メソッドを実行します。ここで，引数「亀」はキャラクタシートにある「亀」を表しています。

なお，「笑顔」は図形クラスのオブジェクトでなければなりません。

6.2.8 「姿を消す」メソッド

・ 書式

オブジェクト変数名.姿を消す

・ 機能

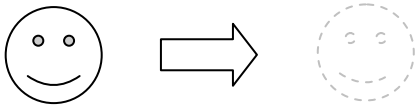
キャラクタの姿を消します。

・ オブジェクト変数名

オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ 使用例

笑顔.姿を消す



キャラクタの姿を消します。

・ 「笑顔」オブジェクトの「姿を消す」メソッドを実行します。これにより，キャラクタは非表示になります。このメソッドを実行するとキャラクタは非表示になりますが，オブジェクトを削除したことにはなりません。
なお，「笑顔」は図形クラスのオブジェクトでなければなりません。

6.2.9 「軌跡をグループ化する」メソッド

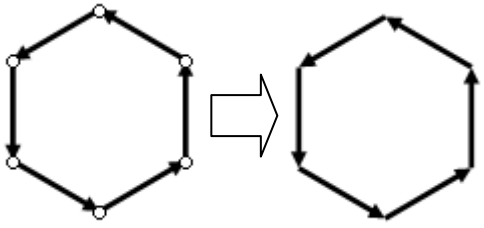
・ 書式

オブジェクト変数名.軌跡をグループ化する 図形名

・ 機能
キャラクタが描いた軌跡の図形をグループ化し，それに図形名を付けます。図形名はグループ化する図形の名前をあらわします。キャラクタが描く軌跡の線の種類が「矢印なしの直線」と「矢印付きの直線」の場合に，「軌跡をグループ化する」メソッドが有効です。詳しくは 6.1.2 をご覧ください。
・ オブジェクト変数名
オブジェクト変数は図形クラスのオブジェクトである必要があります。
・ 図形名「文字列型 (String)」
図形名はグループ化する図形の名前です。
・ 使用例

笑顔.軌跡をグループ化する "図形 1"

キャラクタが描いた軌跡をグループ化し，その名前を「図形 1」にします。



グループ化前　　グループ化後

・ 「笑顔」オブジェクトの「軌跡をグループ化する」メソッドを実行します。これにより，軌跡はグループ化され，その名前を「図形 1」にします。グループ化をすることによって，複数の図形を一つにして扱うことができます。なお，「笑顔」は図形クラスのオブジェクトでなければなりません。

6.2.10 「軌跡を移動する」メソッド

・ 書式

オブジェクト変数名.軌跡を移動する 図形名,X 座標,Y 座標

・ 機能

キャラクタが描いた図形の中心の縦軸座標を X 座標の位置に，横軸座標を Y 座標にそれぞれ移動します。図形名は移動する図形の名前をあらわします。

・ オブジェクト変数名

オブジェクト変数は図形クラスのオブジェクトである必要があります。

・ 図形名「文字列型 (String)」

X 座標「単精度浮動小数点数型 (Single)」，Y 座標「単精度浮動小数点数型 (Single)」

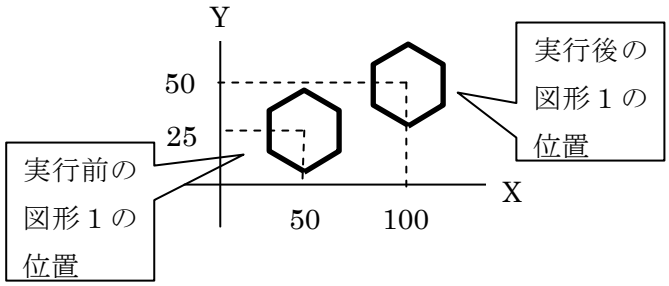
図形名は軌跡が描いた図形の名称，値 1 は X 座標，値 2 は Y 座標です。座標は項目 3.2 を参照してください。

・ 使用例

```
笑顔.軌跡を移動する "図形 1", 100, 50
```

「笑顔」が描いた「図形1」という名前の図形の中心位置をX座標=100 に,Y 座標=50 に移動します。

「笑顔」オブジェクトの「軌跡を移動する」メソッドを実行します。ここで，1 つ目の引数「図形 1」は，「笑顔」が描いた軌跡の図形（ここでは六角形）の名称で，２つ目の引数「100」は「図形 1」の移動後の中心の X 座標を，３つ目の引数「50」は「図形 1」の移動後の中心の Y 座標を表しています。これにより，軌跡の「図形 1」の中心の X 座標が 100 の位置に,Y 座標が 50 の位置にそれぞれ移動します。これ例では，このメソッドを実行する前の図形の中心の座標が（50, 25）の位置にあった場合を表しています。



なお，「笑顔」は図形クラスのオブジェクトでなければなりません。

6.2.11 「軌跡を塗りつぶす」メソッド

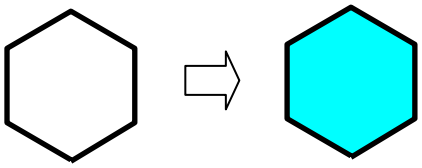
・ 書式

オブジェクト変数名.軌跡を塗りつぶす 図形名，色

- 機能
  キャラクタが描いた軌跡の図形を指定する色で塗りつぶします。
  図形名は塗りつぶす図形の名前をあらわし，色は塗りつぶす色の番号をあらわします。
  キャラクタが描く軌跡の直線が「フリーフォーム」の場合に，「軌跡を塗りつぶす」メソッドが有効です。
  詳しくは 6.1.2 をご覧ください。
- オブジェクト変数名
  オブジェクト変数は図形クラスのオブジェクトである必要があります。
- 図形名「文字列型 (String)」，色「整数型 (Integer)」
  図形名は軌跡が描いた図形の名称，色は塗りつぶす色の番号をあらわしています。
- 使用例

  笑顔.軌跡を塗りつぶす "図形 1", 15

  笑顔が描いた「図形 1」という名前の図形を 15（シアン）の色で塗りつぶします。



実行前　　実行後

- 「笑顔」オブジェクトの「軌跡を塗りつぶす」メソッドを実行します。これにより，笑顔オブジェクトが描いた「図形 1」という名前の図形は 15（シアン）の色で塗りつぶされます。なお，「笑顔」は図形クラスのオブジェクトでなければなりません。

6.2.12 「キャラクタ名を変える」メソッド

- 書式

オブジェクト変数名.キャラクタ名を変える キャラクタ名

- 機能
  キャラクタ名を指定するキャラクタ名に変えます。
- オブジェクト変数名
  オブジェクト変数は図形クラスのオブジェクトである必要があります。
- キャラクタ名「文字列型 (String)」
  キャラクタ名は変更するキャラクタ名をあらわしています。
- 使用例

笑顔.キャラクタ名を変える "太郎"

笑顔のキャラクタ名を「太郎」に変えます。

・ 「笑顔」オブジェクトの「キャラクタ名を変える」メソッドを実行します。これにより，「笑顔」のキャラクタ名を「太郎」に変えます。なお，「笑顔」は図形クラスのオブジェクトでなければなりません。

6.2.13 「三原色を変える」メソッド

・ 書式

オブジェクト変数名.三原色を変える 赤,緑,青

・ 機能
キャラクタの色を赤，緑，青の三原色を合成した色に変えます。
・ オブジェクト変数名
オブジェクト変数は図形クラスのオブジェクトである必要があります。
・ 赤「整数型 (Integer)」，緑「整数型 (Integer)」，青「整数型 (Integer)」
赤，緑，青は三原色の明るさ（濃淡値）をあらわします。
それぞれ 0〜255 の範囲の整数で指定します。
0 が最も暗く，値が大きくなるにつれてその色の明るさが増え，255 で最も明るくなります。
・ 使用例

笑顔.色を変える 255,255,0

笑顔の色をイエロー（赤=255，緑=255，青=0）に変えます。

「笑顔」オブジェクトの「色を変える」メソッドを実行します。
ここで，引数「255,255,0」は赤色成分の明るさ=255，緑色成分の明るさ=255，青色成分の明るさ=0 を表しています。三原色を合成すると，イエローになります。
なお，「笑顔」は図形クラスのオブジェクトでなければなりません。

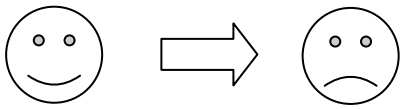
6.2.14 「表情を変える」メソッド

・ 書式

オブジェクト変数名.表情を変える 表情の程度

・ 機能
キャラクタの色を指定した色に変えます。
・ オブジェクト変数名
オブジェクト変数は図形クラスのオブジェクトである必要があります。
・ 表情の程度「単精度浮動小数点数型 (Single)」
表情の程度は口の形をあらわします。0～100 の範囲の値を指定します。
0 は怒顔，100 は笑顔
・ 注意事項
このメソッドは，キャラクタの図形が「スマイル」に対して有効です。
・ 使用例

笑顔.表情を変える 0



キャラクタの表情を 0（怒顔）に変えます。

「笑顔」オブジェクトの「表情を変える」メソッドを実行します。ここで，引数「0」は怒顔を表しています。
なお,「笑顔」は図形クラスのオブジェクトでなければなりません。

6.2.15 「楽器を演奏する」メソッド
・ 書式

オブジェクト変数名.楽器を演奏する 楽譜

・ 機能
オブジェクトの楽器を楽譜にしたがって演奏します。
音楽用プロパティについては 6.15～6.17 をご覧ください。
・ オブジェクト変数名
オブジェクト変数は音楽クラスのオブジェクトである必要があります。
・ 楽譜「文字列型 (String)」
楽譜は演奏する楽譜をあらわしています。
楽譜は①音階，②音符の長さ，③休符の長さ，④音程から構成されています。

①音階

音階は, 表のように, 全角カタカナ, 半角ｶﾀｶﾅ, 全角ひらがなのいずれかで指定します。

ド	レ	ミ	ファ	ソ	ラ	シ
ﾄﾞ	ﾚ	ﾐ	ﾌｧ	ｿ	ﾗ	ｼ
ど	れ	み	ふぁ	そ	ら	し

②音符の長さ

音符は，表のように全角または半角の数字で指定します。音符の長さを指定しない場合は四分音符になります。

全音符	二分音符	四分音符	八分音符	16 分音符	32 分音符
1	2	4	8	A	B
1	2	4	8	A	B

1.5 倍の長さの音符を，付点音符といます。表のように全角または半角の「・」＋「数字またはアルファベット」で指定します。

付点 全音符	付点 二分音符	付点 四分音符	付点 八分音符	付点 16 分音符	付点 32 分音符
・1	・2	・4	・8	・A	・B
･1	･2	･4	･8	･A	･B

③休符の長さ

休符は，表のように全角または半角の「＊」＋「数字またはアルファベット」で指定します。

全休符	二分休符	四分休符	八分休符	16 分休符	32 分休符
＊1	＊2	＊4	＊8	＊A	＊B
*1	*2	*4	*8	*A	*B

1.5 倍の長さの休符を，付点休符といいます。表のように全角または半角の「＊・」＋「数字またはアルファベット」で指定します。

付点 全休符	付点 二分休符	付点 四分休符	付点 八分休符	付点 16 分休符	付点 32 分休符
＊・1	＊・2	＊・4	＊・8	＊・A	＊・B
*･1	*･2	*･4	*･8	*･A	*･B

④音程

音程を次の記号で指定します。

1 オクターブ上がる	1 オクターブ下がる	半音上がる	半音下がる
^	__	#	♭ または b
^	_	#	b

・ 使用例１

奏者.楽器を演奏する "ドドソソララソ２ファファミミレレド＊1"

奏者の楽器を「ドドソソララソ２ファファミミレレド＊1」で演奏します。

「奏者」オブジェクトの「楽器を演奏する」メソッドを実行します。これにより，「奏者」の楽器を「ドドソソララソ２ファファミミレレド＊1」という楽譜で演奏します。「ソ２」は二分音符の「ソ」を，「＊１」は全休符をあらわしています。
なお，「奏者」は音楽クラスのオブジェクトでなければなりません。

・ 使用例２

奏者.楽器を演奏する "どれみふぁそらしど＾"

奏者の楽器を「どれみふぁそらしど＾」で演奏します。

「奏者」オブジェクトの「楽器を演奏する」メソッドを実行します。これにより，「奏者」の楽器を「どれみふぁそらしど＾」という楽譜で演奏します。「ど＾」は半音高い「ど」をあらわしています。なお，「奏者」は音楽クラスのオブジェクトでなければなりません。

・ MIDI 楽器の種類と番号

MIDI 楽器の種類と番号を以下の表にあらわします。

番	楽器	番	楽器	番	楽器	番	楽器
0	生のグランドピアノ	32	生ベース	64	ソプラノサックス	96	シンセ・効果音色１
1	明るい生ピアノ	33	電子ベースギター	65	アルトサックス	97	シンセ・効果音色２
2	電子･グランドピアノ	34	電子ベースギター	66	テナーサックス	98	シンセ・効果音色３
3	ホンキー・トンク・ピ	35	電子ベースギター	67	バリトンサックス	99	シンセ・効果音色４
4	電子・ピアノ１	36	スラップベースギター	68	オーボエ	100	シンセ・効果音色５
5	電子・ピアノ２	37	スラップベースギター	69	イングリッシュ･ホル	101	シンセ・効果音色６
6	ハープシコード	38	シンセ・ベース１	70	バスーン	102	シンセ・効果音色７
7	クラビコード	39	シンセ・ベース２	71	クラリネット	103	シンセ・効果音色８
8	チェレスタ	40	バイオリン	72	ピッコロ	104	シタール
9	グロッケンシュピー	41	ビオラ	73	フルート	105	バンジョー
10	オルゴール	42	チェロ	74	リコーダ	106	三味線
11	ビブラフォン	43	コントラバス	75	パン・フルート	107	琴
12	マリンバ	44	トレモロ・ストリング	76	ビンに息を吹き込む	108	カリンバ
13	シロフォン	45	弦楽ピッチカート	77	尺八	109	バグ・パイプ
14	チューブラベル	46	オーケストラ・ハープ	78	口笛	110	フィドル
15	ダルシマー	47	ティンパニー	79	オカリナ	111	シャナイ
16	電子オルガン	48	弓弦楽器合奏音１	80	シンセ・ソロ音色１	112	ティンカ・ベル
17	電子オルガン	49	弓弦楽器合奏音２	81	シンセ・ソロ音色２	113	アゴゴ
18	電子オルガン	50	合成弓弦楽器合奏音１	82	シンセ・ソロ音色３	114	スティールドラム
19	生の教会オルガン	51	合成弓弦楽器合奏音２	83	シンセ・ソロ音色４	115	ウッド・ブロック
20	生のリード･オルガン	52	合唱	84	シンセ・ソロ音色５	116	和太鼓
21	アコーディオン	53	人声	85	シンセ・ソロ音色６	117	メロディック・タム
22	ハーモニカ	54	合成人声	86	シンセ・ソロ音色７	118	シンセ・ドラム
23	タンゴ･アコーディオ	55	オーケストラ・ヒット	87	シンセ・ソロ音色８	119	リバース・シンバル
24	生ギター	56	トランペット	88	シンセ・多声音色１	120	ギター･フレット･ノ
25	生ギター	57	トロンボーン	89	シンセ・多声音色２	121	ブレス・ノイズ
26	電子ギター	58	チューバ	90	シンセ・多声音色３	122	海辺
27	電子ギター１	59	消音器付き・トランペ	91	シンセ・多声音色４	123	鳥のさえずり
28	電子ギター２	60	フレンチ・ホルン	92	シンセ・多声音色５	124	電話のベル
29	電子ギター３	61	金管楽器・合奏	93	シンセ・多声音色６	125	ヘリコプター
30	電子ギター４	62	合成音・ブラス１	94	シンセ・多声音色７	126	拍手喝采
31	ギター・共鳴音	63	合成音・ブラス２	95	シンセ・多声音色８	127	ガン・ショット

7　プロシージャ

メソッドとは，クラス(オブジェクト)に属している Function プロシージャ(関数)や，Sub プロシージャ(サブルーチン)です。メソッドは決められたクラス（オブジェクト）だけでしか実行することができません。これに対してクラス（オブジェクト）に属していないプロシージャは，プログラム中のどこでも利用できます。ここではクラス(オブジェクト)に属していないプロシージャについて説明します。

プロシージャには，以下のものがあります。カッコ内は項目番号です。

・Function プロシージャ

　キー入力判定(7.1)

・Sub プロシージャ

　待機(7.2)，設定窓表示(7.3)，キャンバス(7.4)，別ファイルに保存 (7.5)，

　別ファイルから読込 (7.6)，図形消去(7.7)，透明化(7.8)，音楽再生(7.9)

7.1 「キー入力判定」Function プロシージャ

・ 書式

キー入力判定(キーコード)

・ 機能

キーコードで指定されるキーが押されているかを判定する。

・ キーコード「長整数型 (Long)」

各キーに割り当てられているキーコードを設定します。キーコードは数値で指定しますが，システムプログラムの中で定義されている定数を用いてもかまいません。

例（その他のキーコードは，以下のキーコード一覧表を参照してください。）

左矢印キー「←」のキーコードは 16 進数で&H25，定数で VK_LEFT

上矢印キー「↑」のキーコードは 16 進数で&H26，定数で VK_UP

右矢印キー「→」のキーコードは 16 進数で&H27，定数で VK_RIGHT

下矢印キー「↓」のキーコードは 16 進数で&H28，定数で VK_DOWN

・ 戻り値「長整数型 (Long)」

戻り値が 0 ならばキーコードで指定されるキーが押されていないことを示します。

戻り値が 0 でなければキーコードで指定されるキーが押されていることを示します。

・ 使用例

```
If キー入力判定(VK_RIGHT) <> 0 Then

右矢印キーが押されているならば
```

・　キーコード一覧表

キーの種類	キーコード	定数	キーの種類	キーコード	定数	キーの種類	キーコード	定数
LeftClick	&H1	VK_LBUTTON	4	&H34	VK_4	5	&H65	VK_NUMPAD5
RightClick	&H2	VK_RBUTTON	5	&H35	VK_5	6	&H66	VK_NUMPAD6
Cancel	&H3	VK_CANCEL	6	&H36	VK_6	7	&H67	VK_NUMPAD7
MiddleClick	&H4	VK_MBUTTON	7	&H37	VK_7	8	&H68	VK_NUMPAD8
BackSpace	&H8	VK_BACK	8	&H38	VK_8	9	&H69	VK_NUMPAD9
TAB	&H9	VK_TAB	9	&H39	VK_9	*	&H6A	VK_MULTIPLY
Clear	&HC	VK_CLEAR	A	&H41	VK_A	+	&H6B	VK_ADD
Enter	&HD	VK_RETURN	B	&H42	VK_B	Enter	&H6C	VK_SEPARATOR
LeftShift	&H10	VK_SHIFT	C	&H43	VK_C	-	&H6D	VK_SUBTRACT
RightShift	&HA1	VK_RSHIFT	D	&H44	VK_D	.	&H6E	VK_DECIMAL
LeftControl	&H11	VK_CONTROL	E	&H45	VK_E	/	&H6F	VK_DIVIDE
RightControl	&HA3	VK_RCONTROL	F	&H46	VK_F	F1	&H70	VK_F1
LeftMenu	&H12	VK_MENU	G	&H47	VK_G	F2	&H71	VK_F2
RightMenu	&HA5	VK_RMENU	H	&H48	VK_H	F3	&H72	VK_F3
Pause	&H13	VK_PAUSE	I	&H49	VK_I	F4	&H73	VK_F4
CapsLock	&H14	VK_CAPITAL	J	&H4A	VK_J	F5	&H74	VK_F5
Esc	&H1B	VK_ESCAPE	K	&H4B	VK_K	F6	&H75	VK_F6
Space	&H20	VK_SPACE	L	&H4C	VK_L	F7	&H76	VK_F7
PageUp	&H21	VK_PRIOR	M	&H4D	VK_M	F8	&H77	VK_F8
PageDown	&H22	VK_NEXT	N	&H4E	VK_N	F9	&H78	VK_F9
End	&H23	VK_END	O	&H4F	VK_O	F10	&H79	VK_F10
Home	&H24	VK_HOME	P	&H50	VK_P	F11	&H7A	VK_F11
←	&H25	VK_LEFT	Q	&H51	VK_Q	F12	&H7B	VK_F12
↑	&H26	VK_UP	R	&H52	VK_R	F13	&H7C	VK_F13
→	&H27	VK_RIGHT	S	&H53	VK_S	F14	&H7D	VK_F14
↓	&H28	VK_DOWN	T	&H54	VK_T	F15	&H7E	VK_F15
Select	&H29	VK_SELECT	U	&H55	VK_U	F16	&H7F	VK_F16
PrintScreen	&H2A	VK_PRINT	V	&H56	VK_V	F17	&H80	VK_F17
Execute	&H2B	VK_EXECUTE	W	&H57	VK_W	F18	&H81	VK_F18
SnapShot	&H2C	VK_SNAPSHOT	X	&H58	VK_X	F19	&H82	VK_F19
Insert	&H2D	VK_INSERT	Y	&H59	VK_Y	F20	&H83	VK_F20
Delete	&H2E	VK_DELETE	Z	&H5A	VK_Z	F21	&H84	VK_F21
Help	&H2F	VK_HELP	0	&H60	VK_NUMPAD0	F22	&H85	VK_F22
0	&H30	VK_0	1	&H61	VK_NUMPAD1	F23	&H86	VK_F23
1	&H31	VK_1	2	&H62	VK_NUMPAD2	F24	&H87	VK_F24
2	&H32	VK_2	3	&H63	VK_NUMPAD3	NumLock	&H90	VK_NUMLOCK
3	&H33	VK_3	4	&H64	VK_NUMPAD4			

7.2 「待機」Sub プロシージャ

・ 書式

```
待機 待機時間
```

・ 機能

待機時間だけプログラムの実行を中断します。

前述の「待ち時間」プロパティはプログラムの終了まで各動作の間に「待ち時間」で設定した時間が挿入されますが，「待機」Sub プロシージャは，プロシージャを実行した時点だけプログラムの実行を中断します。詳しくは 6.1.1 をご覧ください。

・ 待機時間「長整数型 (Long)」

待機時間をミリ秒（ms）単位で与えます。

・ 使用例

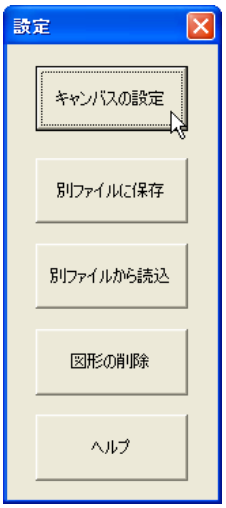
```
待機 50
```

50 ミリ秒待機します。

7.3 「設定の窓表示」Sub プロシージャ

・ 書式

```
設定の窓表示
```



・ 機能

設定のためのユーザフォームを表示します。

詳しくは 8 をご覧ください。

・ 使用例

```
設定の窓表示
```

設定のためのユーザフォームを表示します。

7.4 「キャンバス」Sub プロシージャ

・ 書式

```
キャンバス
```

・ 機能

キャンバスを設定するためのユーザフォームを表示します。

詳しくは 8.1 をご覧ください。

・ 使用例

キャンバス

キャンバスを設定するためのユーザフォームを表示します。

7.5 「別ファイルに保存」Sub プロシージャ

・ 書式

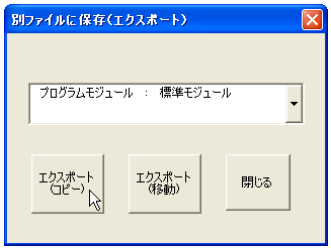
別ファイルに保存

・ 機能

プログラムモジュールを別ファイルに保存するためのユーザフォームを表示します。

詳しくは 8.2 をご覧ください。

・ 使用例



別ファイルに保存

別ファイルに保存するためのユーザフォームを表示します。

7.6 「別ファイルから読込」Sub プロシージャ

・ 書式

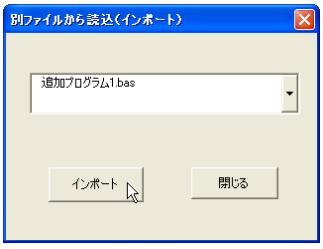
別ファイルから読込

・ 機能

別ファイルからプログラムモジュールを読み込むためのユーザフォームを表示します。

詳しくは 8.3 をご覧ください。

・ 使用例



別ファイルから読込み

別ファイルにから読み込むためのユーザフォームを表示します。

7.7 「図形消去」Sub プロシージャ

・ 書式

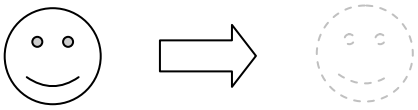
図形消去

・ 機能

ワークシート上のキャラクタが描いたすべての図形を消去します。

・ 使用例

図形消去



ワークシート上のキャラクタが描いたすべての図形を消去します。

7.8 「透明化」Sub プロシージャ

・ 書式

透明化 透明度

・ 機能

ユーザフォームを透明度に従って透明化します。

・ 透明度「長整数型 (Long)」

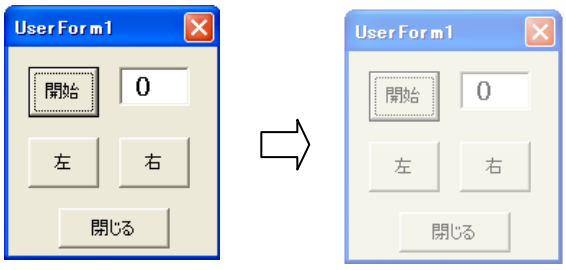
0～100%の値を設定します。この値が小さいほど透明に近くなります。

・ 注意事項

UserForm_Activate イベントプロシージャの中で使用してください。

・ 使用例

```
Private Sub UserForm_Activate()
    透明化 50
End Sub
```



透明化前　　透明化後

ユーザフォームを 50%の透明度で透明化します。

7.9 「音楽再生」Sub プロシージャ

・ 書式

音楽再生 音声ファイル名

・ 機能

音声ファイル名で示される音声を再生します。

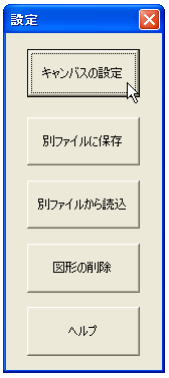
・ 音声ファイル名「文字列型 (String)」

再生する wav 形式の音声のファイル名を指定します。このファイルは SmileProgram や SmileClass と同じフォルダに入れておきます。なお，SmileProgram や SmileClass が入っているフォルダ内に見本の音声ファイルとして oto1.wav と oto2.wav が入っています。

・ 使用例

```
音声再生 "oto1.wav"
```

「oto1.wav」という名前の音声ファイルを再生します。



8 設定

上記 2.5 のプログラムの実行で「設定」を選択すると，図のような「設定」用の窓が開きます。

ここから，「キャンバスの設定」，「別ファイルから読込」，「別ファイルに保存」，「図形の削除」，「ヘルプ」の操作ができます。

詳細を以下に説明します。

8.1 キャンパスの設定

「キャンパスの設定」のボタンをクリックすると，図のような「キャンパスの設定」用の窓が開きます。

窓には，「横位置」スクロールバー，「縦位置」スクロールバー，「倍率」スクロールバー，「全リセット」ボタン，「網目のセット」ボタン，「網目のリセット」ボタン，「目盛表示」ボタン，「目盛消去」ボタンのコントロールが配置されています。これらのコントロールを操作してキャンバスの設定を行います。

詳細を以下に説明します。

8.1.1 「横位置」スクロールバー

このスクロールバーを操作すると，座標を横方向にスクロールすることができます。横位置のテキストボックスに数値が表示されますが，ここに直接数値を入力することはできません。画面の中央に縦座標軸を配置するとテキストボックスには「0」が表示されます。

8.1.2 「縦位置」スクロールバー

このスクロールバーを操作すると，座標を縦方向にスクロールすることができます。

縦位置のテキストボックスに数値が表示されますが，ここに直接数値を入力することはできません。画面の中央に横座標軸を配置するとテキストボックスには「0」が表示されます。

8.1.3 「倍率」スクロールバー

このスクロールバーを操作すると，座標を拡大・縮小することができます。

倍率%のテキストボックスに数値が表示されますが，ここに直接数値を入力することはできません。最初の大きさに戻すとテキストボックスには「100」が表示されます。

8.1.4 「位置 0　倍率 100」ボタン

横位置と縦位置をそれぞれ 0 の位置に，倍率を 100%に戻します。このボタンをクリックしても，位置 0，倍率 100 にならない場合は再度クリックしてください。

8.1.5 「キャンバスのセット」ボタン

列幅を 10 ピクセル，行高を 10 ピクセルにします。また，座標軸を表示します。

8.1.6 「キャンバスのリセット」ボタン

列幅を 72 ピクセル，行高を 18 ピクセルにします。また，座標軸を消去します。

8.1.7 「座標軸の目盛表示」ボタン

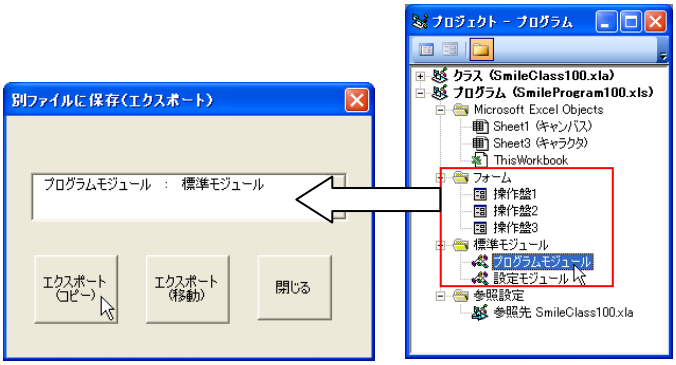
このボタンをクリックすると，座標軸に目盛を表示します。

8.1.8 「座標軸の目盛消去」ボタン

このボタンをクリックすると，座標軸の目盛を消去します。

8.2 別ファイルに保存

プロジェクトエクスプローラに表示されている標準モジュールやフォームモジュール内のモジュールを別ファイルにして「SmileProgram1.00.xls」や「SmileClass1.00.xla」が入っているフォルダ内に保存します。



標準モジュール内のモジュールは，拡張子が「bas」のファイルにして保存し，フォームモジュール内のモジュールは，拡張子が「frm」と「frx」のファイルにして保存します。「bas」は Basic の略称で，ユーザが記述したソースコードで構成されています。また，「frm」は Form の略称で，「frx」は Form Extension の略称です。

「bas」ファイルは，ユーザが記述したソースコードで構成されており，「frm」ファイルは，コントロールのレイアウトに関する情報と，ユーザが記述したソースコードで構成されています。保存したファイルのうち，「bas」と「frm」はテキストエディタ等のソフトウェアで開き編集することができます。

8.2.1 エクスポート（コピー）

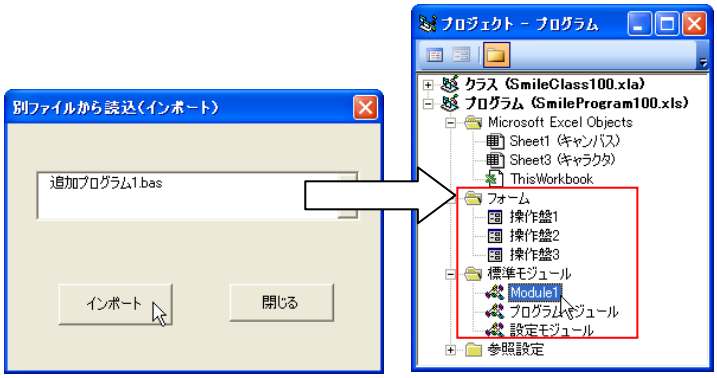
このボタンをクリックすると，「プロジェクトエクスプローラに表示されている標準モジュールやフォームモジュール内のモジュールの複製を別ファイルにすることができます。

8.2.2 エクスポート（移動）

このボタンをクリックすると，「SmileProgram1.00.xls」内にある標準モジュールやフォームモジュール内のモジュールを別ファイルとして移動することができます。

8.3 別ファイルから読み込み

拡張子が「bas」のファイルを標準モジュールに読み込み，拡張子が「frm」と「frx」のファイルをフォームモジュールに読み込みます。拡張子が「bas」のファイルを読み込むと，モジュール名は「Module＊」となります。「＊」は連番です。



なお，「bas」，「frm」，「frx」の各ファイルは「SmileProgram1.00.xls」や「SmileClass1.00.xla」が入っているフォルダ内に入っている必要があります。

8.4 図形の削除

「キャンパス」シート上のすべての図形を削除します。

8.5 ヘルプ

注意事項が書かれた窓を表示します。

8.6 サンプルプログラムについて

Smile1.00 のプログラムモジュールに 17 種類のサンプルが入っています。

サンプル１：極座標形式で矢印付きの直線で六角形を描画
サンプル２：極座標形式で矢印なしの直線で六角形を描画
サンプル３：極座標形式でフリーフォームで六角形を描画，塗りつぶし、図形の移動
サンプル４：極座標形式で矢印付きの直線で六角形を描画，キャラクタは亀
サンプル５：絶対座標形式で矢印なしの直線で六角形を描画
サンプル６：相対座標形式で矢印なしの直線で六角形を描画
サンプル７：極座標形式で矢印なしの直線で星を描画
サンプル８：絶対座標形式で矢印なしの直線でサインカーブを描画、図形の移動
サンプル９：オブジェクトの代入、キャラクタの拡大・縮小
サンプル 10：オブジェクト配列変数、キャラクタの拡大・縮小
サンプル 11：軌跡の描画
サンプル 12：シューティングゲーム

サンプル 13：シューティングゲーム（→と←と ESC キーで操作）
サンプル 14：音楽の演奏
サンプル 15：スマイルの表情と色を変える
サンプル 16：スマイルの表情と三原色を変える
サンプル 17：「Addition1.bas」をインポートする　うなりの描画

8.7 スマイルについて

・ 構成ファイル

SmileProgram1.00.xls	プログラムファイル
SmileClass1.00.xla	アドインクラスファイル
Addition1.bas	追加プログラムファイル(インポート用見本ファイル)
oto1.wav	音声ファイル 1(ゲーム用)
oto2.wav	音声ファイル 2(ゲーム用)
Smile1.00.txt	ヘルプファイル
Smile1.00.pdf	取り扱い説明書

・ 履歴

2008.6.29　Ver1.00　　初版(日本情報科教育学会第 1 回全国大会にて公開)

・ 作者　東京都立日比谷高等学校　天良和男 (TENRA Kazuo)

・ 著作権

このプログラムはフリーソフトウェアですが，著作権は放棄しておりません。

・ 免責事項

本プログラムを使用したことで発生したいかなる不都合に対して，作者は責任を負いかねますのであらかじめご了承ください。

・ 再配布

現時点では再配布は認めておりません。

・ 入手方法

スマイルは以下のページからダウンロードできます。

http://homepage3.nifty.com/tenra/

修正プログラムについても上記のページにアップする予定です。